AFRL-IF-RS-TR-2000-68
Final Technical Report
May 2000

# AN INTERACTIVE ASSISTANT FOR DECISION MAKING

University of Rochester

James F. Allen and George M. Ferguson

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*



AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-68 has been reviewed and is approved for publication.

APPROVED: William E. Rzepka

WILLIAM E. RZEPKA
Project Engineer

FOR THE DIRECTOR: Northrup Fowler

NORTHRUP FOWLER, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTD, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

| REPORT DOCUMENTATION PAGE | Form Approved OMB No. 0704-0188 |
|---|---|



**1. AGENCY USE ONLY** *(Leave blank)*

**2. REPORT DATE**
MAY 2000

**3. REPORT TYPE AND DATES COVERED**
Final Sep 97 - Nov 99

**4. TITLE AND SUBTITLE**
AN INTERACTIVE ASSISTANT FOR DECISION MAKING

**5. FUNDING NUMBERS**
C - F30602-97-1-0348
PE - 62702F
PR - 5581
TA - 27
WU - 03

**6. AUTHOR(S)**
James F. Allen and George M. Ferguson

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
University of Rochester
Computer Science Department
734 Computer Studies Building
Rochester NY 14627-0226

**8. PERFORMING ORGANIZATION REPORT NUMBER**
N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Office of Naval Research
Boston Regional Office
495 Summer St., Rm 103
Boston MA 02210-2108

Air Force Research Laboratory/IFTD
525 Brooks Road
Rome NY 13441-4505

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
AFRL-IF-RS-TR-2000-68

**11. SUPPLEMENTARY NOTES**
Air Force Research Laboratory Project Engineer: William E. Rzepka/IFTD/(315) 330-2762

**12a. DISTRIBUTION AVAILABILITY STATEMENT**
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*
This project involved extending and modifying our previous work on interactive decision-making in TRIPS, the Rochester Interactive Planning System (Ferguson and Allen, 1998; Ferguson, et.al., 1996). The goal was to enable information access to and from remote sources and seamlessly integrate them into the overall system. In this project, we have focused on using Java to facilitate interfacing TRIPS to remote users and, eventually, for integrating additional information sources and reasoners into the system. Java offers potential advantages at several different levels, from the portability of the graphical components, to the simplified networking, all the way to tightly-coupled, object-oriented, method-call inter-module communication. This project was intended to investigate these possibilities, use as many as seem useful and feasible, and provide feedback on the overall suitability of Java in a system like TRIPS. The main goals of the project were (a) evaluation of Java for use in TRIPS; (b) porting TRIPS interface components to Java to enable remote access to the system; and (c) porting to and evaluation of a Fujitsu 1200 tablet computer with wireless network to enable truly portable access to the system.

**14. SUBJECT TERMS**
Interactive Decision Making, Remote Access

**15. NUMBER OF PAGES**
102

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

# Abstract

This project involved extending and modifying our previous work on interactive decision-making in TRIPS, The Rochester Interactive Planning System (Ferguson and Allen, 1998; Ferguson, et. al., 1996). The goal was to enable information access to and from remote sources and seamlessly integrate them into the overall system. In this project, we have focused on using Java to facilitate interfacing TRIPS to remote users and, eventually, for integrating additional information sources and reasoners into the system. Java offers potential advantages at several different levels, from the portability of the graphical components, to the simplified networking, all the way to tightly-coupled, object-oriented, method-call inter-module communication. This project was intended to investigate these possibilities, use as many as seem useful and feasible, and provide feedback on the overall suitability of Java in a system like TRIPS. The main goals of the project were (a) evaluation of Java for use in TRIPS; (b) porting TRIPS interface components to Java to enable remote access to the system; and (c) porting to and evaluation of a Fujitsu 1200 tablet computer with wireless network to enable truly portable access to the system.

# Table Of Contents

## List of Figures

# 1 TRIPS Overview

TRIPS, The Rochester Interactive Planning System (Ferguson and Allen, 1998) is the latest in a series of prototype collaborative planning assistants developed at the University of Rochester's Department of Computer Science (Allen et al., 1995; Ferguson, Allen, and Miller, 1996; Ferguson et al., 1996). The goal of the project is an intelligent planning assistant that interacts with its human manager using a combination of natural language and graphical displays. The two of them collaborate to construct plans in crisis situations. The system understands the interaction as a dialogue between it and the human. The dialogue provides the context for interpreting human utterances and actions, and provides the structure for deciding what to do in response. With the human in the loop, they and the system together can solve harder problems faster than either could solve alone.

TRIPS operates in a simplified logistics and transportation world, with cargos being delivered using a variety of vehicles. One example scenario involves evacuating the island of Pacifica (see Figure 1) ahead of an approaching hurricane. The manager's task is to plan the evacuation, using a variety of vehicles (with varying capabilities) at his or her disposal. There may be a variety of constraints placed on the final plans, such as time, cost, weather effects, and so on.

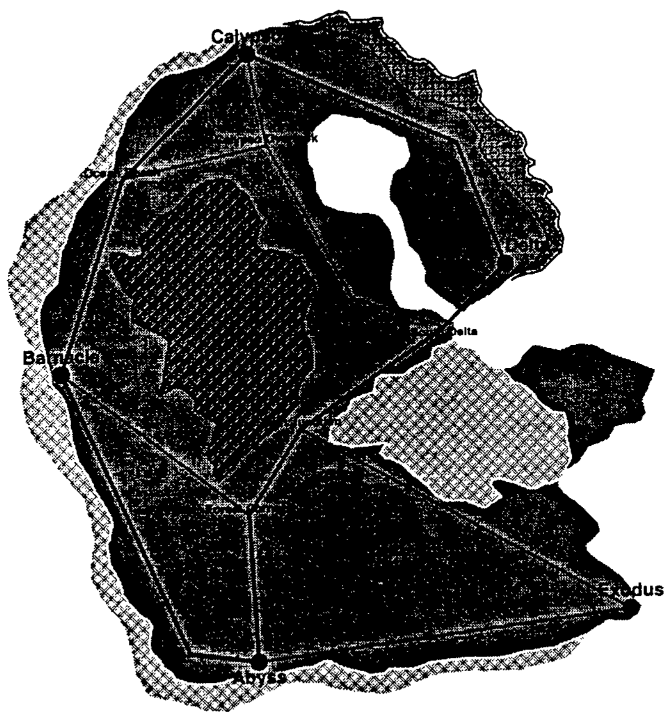

**Figure 1: Map of Pacifica**

TRIPS is designed as a set of loosely-coupled modules that exchange information by passing KQML (Finin et al., 1993) messages. A schematic description of the system is shown in Figure 2. At the top of the schematic are modality processing modules, such speech recognition and generation, keyboard input and output, and interactive graphical displays. Input from these modules is parsed into a uniform representation of the user's input as one or more communicative acts.

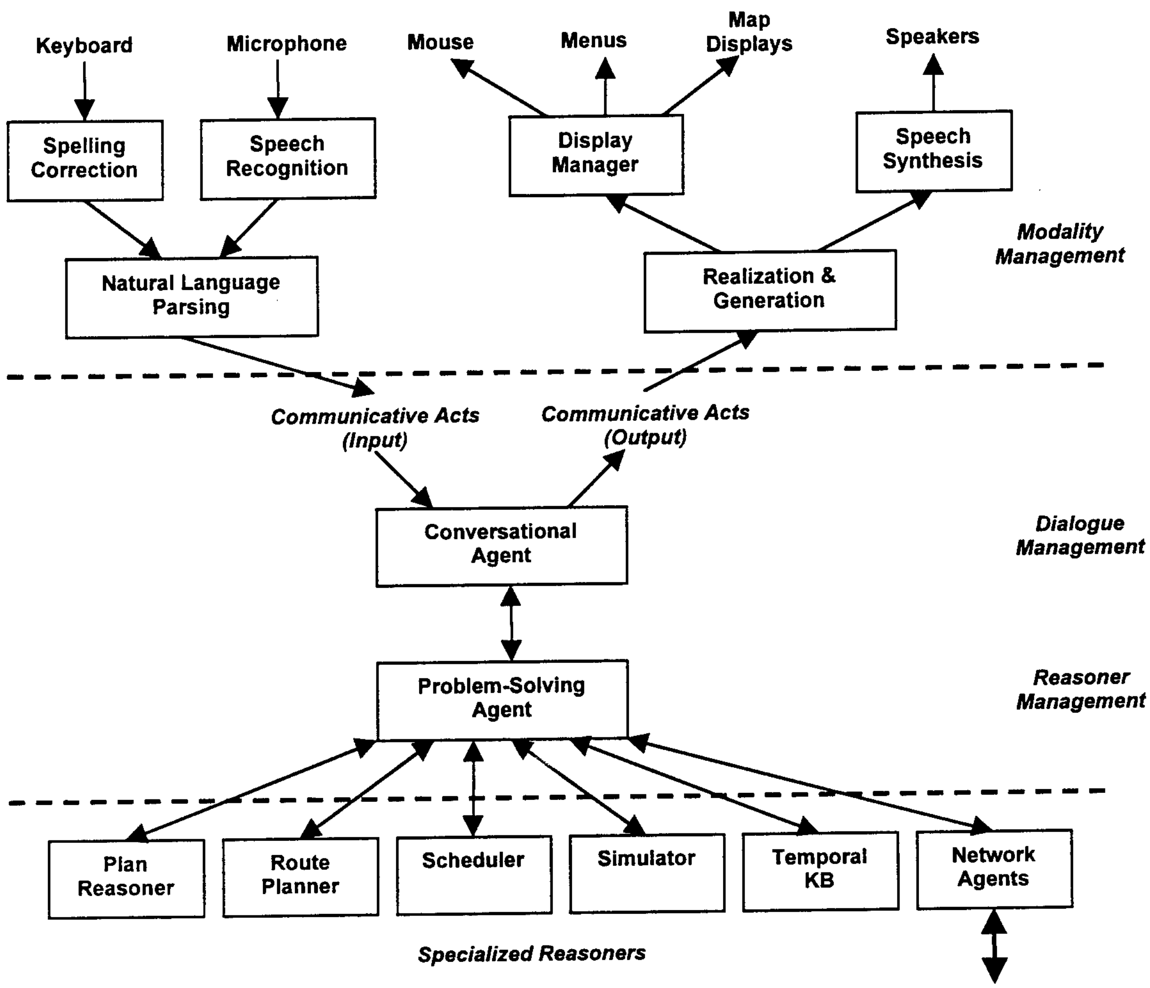


**Figure 2: TRIPS Architecture**

The middle layer in the TRIPS architecture contains the core modules of the system, responsible for mantaining the conversation with the user and helping them achieve their (and the system's) objectives. The Conversational Agent combines the interpreted communicative acts from the input with the discourse context in order to determine the intended speech acts, which might be either indirect ("Do you know the time?") or ambiguous ("Send the truck to Delta" when there are two trucks). The Problem-Solving Manager plays two roles in maintaining the dialogue. First, it helps resolve ambiguities by applying plan recognition techniques. In the previous example of an ambiguous reference to "the truck," for example, the PSM might infer that only one truck is not already at Delta, and so the user must be referring to it. Second, it coordinates the invocation of the specialized reasoners that provide solutions in service of user and system objectives.

These specialized reasoners form the bottom layer of the TRIPS architecture, and currently include a powerful but incomplete temporal logic-based planner, router, scheduler, temporal knowledge base, and a fast simulator with data mining capabilities for detecting (and hopefully correcting) problems with planned activities. The Problem-Solving Manager invokes these reasoners as appropriate, and integrates their responses into the problem-solving context.

Finally, the Conversational Agent uses the results of task-specific problem-solving (e.g., a new part of a plan, or an answer to a query) together with general dialogue principles to determine appropriate responses. Both spoken language and graphical displays can be generated from the intended communicative acts specified by the Conversational Agent.

In addition to the components shown in Figure 2, TRIPS relies on extensive infrastructure support the message-passing communication, process management, logging, and debugging. In particular, the message-passing is implemented using a socket-based hub topology controlled by a central Facilitator (formerly Input Manager) module. This component provides naming services (registration and lookup), performs complete KQML syntax validation, and supports several types of broadcast used to disseminate information among TRIPS components. In conjunction with its logging capabilities, it supports real-time session replay the message traffic. Another component, the Process Manager, connects components to the Facilitator and provides process control and status checking services. The Process Manager allows any program that can read standard input and write standard output to be connected into the TRIPS communication infrastructure without any additional coding.

## 2 Java Reimplementation Project

In this project, we have investigated the use of the Java programming language in the development of TRIPS, to enable remote access by users and to available resources. The question "why Java?" is worth asking, since there is so much hype about Java these days that just about everything is either "Java-ready" or "Java-enabled" or what have you. For us, there are really four benefits, two general and two more specific:

1. **Platform-independence**: Java holds out the promise of being a truly portable language from the low-level programming details to the high-level look-and-feel issues. Whether this promise is realized has yet to be seen, but initial Java implementations and our experiments with them are promising. (When we started this project, we were using version 1.1 of the Java Development Kit (JDK). At the end of the project, the latest version is 1.3.)
2. **Ease of Programming**: All hype to the contrary, programming at the level required for TRIPS components will never be easy. However, Java is more sanely designed than C, less confusing than C++, and less prone to error than Perl, its closest platform-independent competitor. It also incorporates impor-

tant and powerful features from Lisp (notably garbage collection and reflection) that make large-scale, highly-dynamic programs feasible.

3. **Support for Graphics**: An important part of TRIPS is the combination of graphical interaction and language. The Java Abstract Windowing Toolkit, the Swing classes (a.k.a. JFC, the Java Foundation Classes), and recent developments such as the Java Media Framework (JMF) provide platform-independent, relatively easy to use tools for creating graphical and multimodal user interfaces.

4. **Support for Networking**: Java provides the most convenient interface to network programming of any language we have investigated. It also provides the opportunity for us to refine the model of inter-module communication in TRIPS to use object-oriented methods where these are appropriate (see below).

5. **Support for Security**: Although security issues are not a principal concern of our work, Java provides a powerful, well-thought-out security model for the development of distributed applications. By using Java, we can easily leverage this support should security issues becomes significant in the future. The ongoing development of the Java security model is an open process, thereby ensuring that the best solutions are designed, tested, and deployed. This is in marked constrast to some of the proprietary alternatives.

As described in the previous section, TRIPS is already a fully distributed, heterogeneous system. This will not change in the forseeable future, since Java is not the right tool for every task, and TRIPS has modules that perform a wide variety of different tasks. For example, Java will probably never execute efficiently enough to use for online speech recognition, nor is it likely to provide the combination of tools and efficient compilation that we get from implementing some of the knowledge-based modules in Lisp. In situations where graphical displays or networking are signficant aspects of a module's function, however, Java should provide a uniform solution to the issues involved in implementing such modules effectively.

# 3 Development Plan

Our plan was to approach the development of a version of TRIPS based on Java in two phases, in addition to peforming an ongoing evaluation of the pros and cons of using Java.

The first phase was a redesign and reimplementation of the interface components of TRIPS using Java. These modules include the Keyboard Manager, Speech Controller, Audio Control Panel, Transcript, Map Viewer, and Plan Construction Window. This provided the following benefits:

1. We were able to familiarize ourselves with the Java language and the graphical and networking aspects in particular.

2. We were able to evaluate whether Java, in its current state of development, can support the type of applications we need in TRIPS and, if not, whether it may in the future and we should wait, or whether we should change the de-

sign of those applications, or whether Java simply isn't a good option for some applications.

3. We were able to develop tools that can be shared among modules, for example, classes that make inter-module (KQML) communication easy.

4. We were able to use this opportunity to resolve outstanding problems in current TRIPS components, add new functionality where needed, and rationalize the implementation in places where it had gotten too *ad hoc*.

During this first phase of the project, we retained the underlying TRIPS communication architecture based on modules exchanging KQML messages via the Facilitator (Input Manager). We also retained the idea that modules basically read messages from an input stream (typically derived from their standard input) and print messages to an output stream (typically derived from their standard output). Java makes several aspects of these operations simpler than they were in C or Perl, however, as will be described in more detail in the next section.

Finally, several important components of TRIPS, such as the Discourse Manager, Problem Solver, and Planner, were not changed. These modules are "faceless" computation engines, and Java is probably not the appropriate tool for them, as we noted in the previous section. The result of the first phase of the project, once completed, will be a version of TRIPS that can be used from any Java-enabled platform, although it will need access to the other, non-Java components of the system over the network.

Our focus in this project was on that first phase. However, once the first phase is complete, a second phase will look using Java's object-oriented model to effect a much tighter integration of the TRIPS modules. Specifically, this means:

5. Explicit message-based communication will be replaced by Java Remote Method Invocation (RMI) calls. That is, rather than sending a message requesting that a module perform some service, a module can simply invoke the appropriate method on an appropriate object, and Java will look after the distributed nature of the computation.

6. Modules not written in Java will get Java wrappers that use the Java Native Interface (JNI) to access their functionality.

This phase of the project represents a much more radical change to the TRIPS architecture, and it is not yet clear that this the way to go. In the first place, the functionality of several components of TRIPS simply do not fit the object-oriented, method-call framework. For example, when the speech recognizer has recognized a new word, it wants to simply broadcast that fact to the world, at least conceptually. Several other components have a similar flavour, which is derived from the AI notion of a "blackboard system," where "interesting" results and requests are posted on a blackboard shared by all modules, and modules "fire" when they see what they need on the blackboard. It might be possible to implement this using Java (after all, the current Facilitator already maps "broadcasts" to "interested" modules).

On the other hand, other aspects of TRIPS would be much easier to implement using the RMI model. In cases where a request and reply really are functional, in the sense that the caller wants the answer before proceeding, it is much easier to simply make a method call than to send the message, setup state for when the reply is received, wait for the reply, then try to restore state when it arrives. In fact, the current version of TRIPS does this poorly, and this is an architectural bottleneck to further development of the system.

Finally, this second phase of Java redesign has some corollary implications for the TRIPS infrastructure. For example, it has proven invaluable to have a complete log of all messages exchanged in a session. This allows us to reply a session without running the back-end reasoners, not to mention being able to feed the messages back to a module to get it into the right state for debugging. It is not clear how this would work in the RMI model, although some kind of classes for transparently logging inter-module communication could perhaps be developed. In any case, supporting capabilities such as session replay would be challenging.

The ultimate benefit of the second phase of the TRIPS redesign would not be that the entire system could run on a Java-enabled platform. As noted above, there will probably always be components that are not written in Java, for various reasons. However, if it was successful, the entire TRIPS system would then be open to object-oriented interaction with other network services, such as through a CORBA or COM interface. This would allow other systems to use TRIPS services, and probably more importantly for us, would allow TRIPS to use other data sources and services more transparently.

# 4 Accomplishments

Our accomplishments to date in the first phase of our reimplementation project consist in porting interface components of the TRIPS system to Java. The new components developed in this phase of the effort are now part of the TRIPS core and have been successfully demonstrated numerous times. These include demonstrations at the 1998 and 1999 AAAI Intelligent Systems Demonstrations program, at meetings of both the ARPA-Rome Labs Planning Initiative (ARPI) and the DARPA Control of Agent-Based Systems (CoABS) programs, at the 1999 Rome Labs Scientific Advisory Board review, and of course many demonstrations for visitors and press in our labs at Rochester.

Throughout this effort, we have been compiling a list of issues involved in the migration to Java. These include things that should work but don't, things that work but look different, things that we need to do differently, and things that we probably can't do at all. The Java components are being tested on Unix, Windows 95, Windows NT, and Macintosh platforms. The remainder of this section highlights some of the accomplishments.

## 4.1 Java KQML Infrastructure

We have developed a set of Java classes for reading, representing, and sending KQML messages. These are used in the new modules, and will be developed as necessary to support additional TRIPS components. Javadoc documentation of these classes is presented in Appendix A.

## 4.2 Basic Interface Components

We have completed the reimplementation of the Keyboard, Transcript, and Speech Controller modules. These modules were originally written in C, C++, or Perl/Tk and rely on the X Window System directly for their graphical displays. The emphasis here was on the graphical aspects of Java, and whether we could make the modules look and feel the way they should. In general the answer is yes, although there are some differences between what we can do as a (well-behaved) Java application (or applet; see below) and what we could do as a full-fledged X Windows application.

For example, with raw Xlib we can "grab" the X server and receive notification of keypress events even when the mouse pointer is outside any of the TRIPS windows. This has proven useful in controlling the speech recognizer from the keyboard while using the mouse pointer for gesturing (multimodal input). There is no way to support this (portably) in Java. We are therefore forced to develop a new approach to handling multimodal input, essentially by moving to a continuous-listening model of speech recognition.

## 4.3 TRIPS Facilitator (Input Manager)

We have reimplemented the TRIPS Facilitator (a.k.a. Input Manager) itself, as a test of the networking support in Java, a test of our KQML handling, and a good stress test of Java overall. It was not clear at the outset whether Java would be fast enough to handle the message traffic, nor whether it could display the message traffic as the current Input Manager does (which requires some fairly intensive drawing). Initial results are very promising.

Figure 3 shows a raw throughput comparison between the original C implementation of the Facilitator and the new Java reimplementation. Times given are averages over ten trials of the total time for one client to send the given number of messages to one receiving client via the Facilitator, with all logging and display enabled (as would be used in the running system). It is clear that the Java implementation clearly outperforms the C implementation except when a very small number of messages are exchanged, where the cost of starting the Java runtime dominates the cost of the message-passing. Further, the Java implementation is scaling significantly better as the amount of message traffic increases. We were (pleasantly) surprised by these results. We believe that the Java implementation does better because of more efficient I/O operations—the C implementation makes repeated single-character calls to `read()` (necessary to properly parse incoming asynchronous KQML), while the Java I/O classes and the use of multiple reader threads saves the overhead of the system calls.

Figure 4 shows some other statistics regarding the Java reimplementation of the TRIPS Facilitator. These times are averages over ten trials to send and receive 10,000 messages under different configurations. The first category is the same as the rightmost category of Figure 3, namely the default configuration with both logging and display enabled. The second and third categories show the effect of disabling the log and disbling both the log and the display, respectively. For neither the C nor the Java implementation do these have a large effect on system throughput, which is a very positive result. The final category in Figure 4 shows how the performance differs if ten clients are sending to each other in a round-robin fashion, rather than the one-to-one configuration used in all the other trials. The results show that there are no adverse effects from the repeated registry lookups (we expected this, but there might have been cache effects).

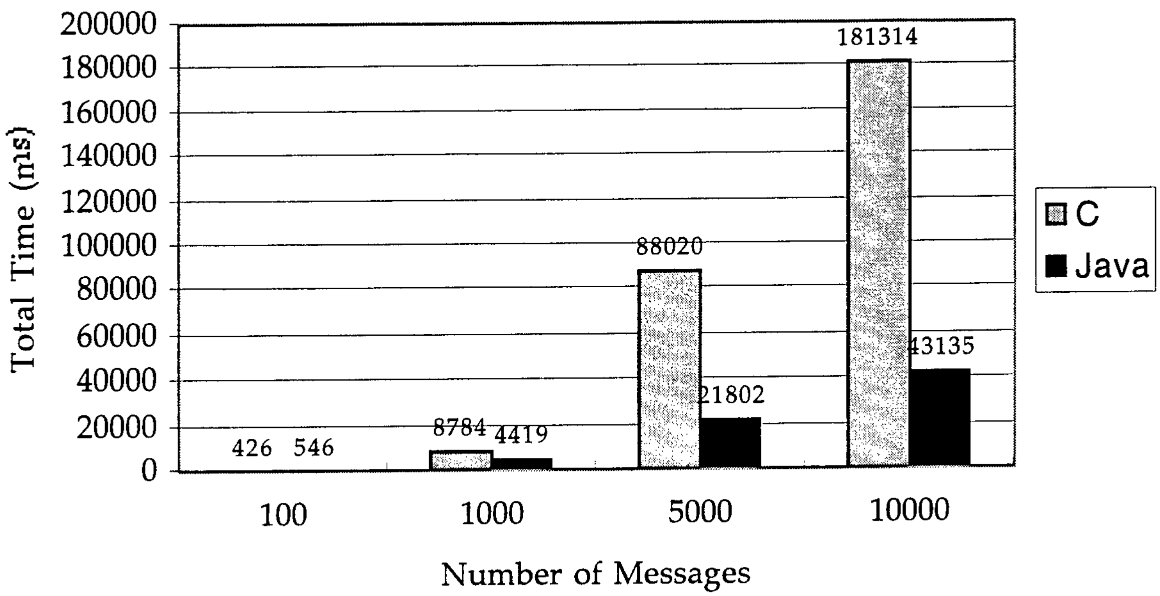


**Figure 3: Throughput comparison of C and Java Facilitators**

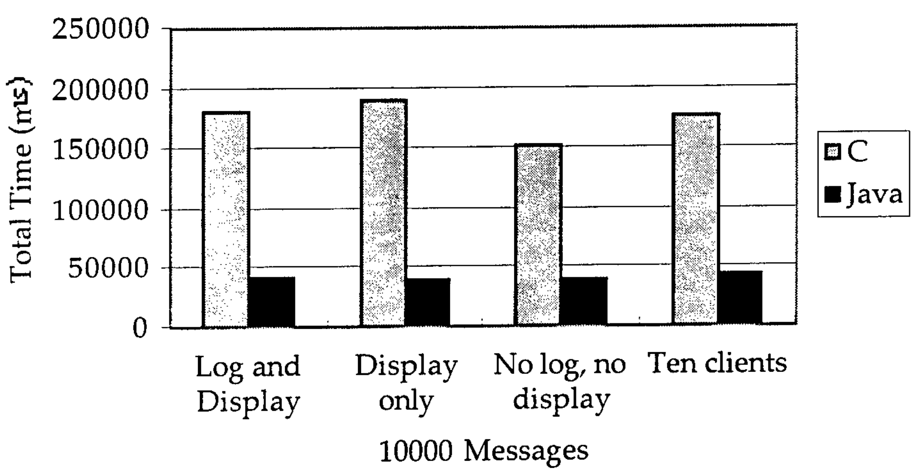


**Figure 4: Effects of logging, display, and registry operations on Facilitator throughput**

## 4.4 Applets and the "Java Desktop"

Java supports both standalone applications and lighter-weight "applets" that run in the context of a larger application (typically a web browser). This section briefly describes our work towards the design and implementation of TRIPS components as applets that run on a "Java Desktop" on any Java-enabled platform. The goal of this part of the project is to allow TRIPS to be accessed from their web browser. The interface components started on this Java Desktop would be connected to the rest of the TRIPS system using standard network connections to the TRIPS Facilitator (Input Manager). Since not all components have yet been ported to Java (and, indeed, it seems unlikely that some of them ever would be), it is important that Java support the existing TRIPS communication architecture.

As a first feasibility test of Java applets for TRIPS interface components, we converted our Keyboard Manager module, a Java application, into an applet. This conversion is fairly straightforward, but we have developed tools to make it even easier in the future. As a practical matter, it is very convenient to be able to run TRIPS components as either applications or applets. In particular, it is easier to debug standalone Java applications than to debug applets running in a browser that was not designed for their debugging. The `TripsApplet` classes we developed to support this dual usage are documented in Appendix B. The result of this work is that all interface components written in Java can be invoked as either applications or applets. As new components are developed using Java, they will automatically have this functionality also.

Three issues are worth mentioning regarding the use of applets in TRIPS (and more generally).

1. First, it is really very simple to do. The boilerplate included in Appendix B, Section B.3, can be used to turn any applet into an application. Our support classes look after hiding many of the differences between applets and applications, such as where their invocation parameters come from (`PARAM` tags for applets, command-line arguments via the `String[]` parameter to the `main` method for applications).
2. Second, while Java promises write-once, run-anywhere code, and Java-enabled browsers promise a Java environment on any platform, attempting to use a browser as one's Java environment is risky business. The main reason is that browser development typically lags the development of Java as a whole. Recent, often powerful, features of Java are typically missing from the version of Java supported by a browser. Netscape is particularly bad in this respect; Internet Explorer at least has a provision for using another installed Java Runtime Environment (JRE). The solution we have found is to use the so-called "Java Plug-in" available from Javasoft, which allows applets to run using another JRE installed on the machine. This way applets can be run against the latest version of the JRE available from Javasoft. Eventually one can hope that this situation will improve, but meanwhile there is still this one requirement on clients that would hope to access the Java Desktop from their browser.
3. Finally, there are security issues. Specifically, applets run from a browser are flagged as "insecure" by the browser. These insecure applets are displayed

with a (deliberately) distracting titlebar, and they lack some of the capabilities of applications. We therefore investigated the use of so-called "signed" applets for use with TRIPS, both to preserve the aesthetic quality of the user interface and to enable the socket-based connections on which the TRIPS communication infrastructure depends. At this point, it is safe to say only that the situation is in flux. Our notes on signing an applet (providing it with an encrypted authorization) in the Java 1.1 model are presented in Appendix C. Towards the end of this project, Java 1.2 (a.k.a. "Java2") was released, and promised a much cleaner implementation of the applet signing process.

In the end, the results of our initial efforts with applets for TRIPS were very promising. Their functionality was essentially identical to applications, including tricky issues like window placement and supporting multiple frames. Running multiple applets rather than multiple applications avoids starting several large Unix processes, one for each Java Runtime Environment. (On the other hand, actually obtaining that benefit requires an efficient implementation of threads in the JRE, something which until fairly recently was not reliably available.)

## 4.5 Remote Audio Using Java Sound

Audio has always been a problem for TRIPS. It is, of course, essential for a speech-based interactive system to be able to receive audio input from the user for use in speech recognition and to produce audio output in the form of system-generated speech. Unfortunately, whereas the X Window System is a mature, freely-available system for networked graphics, there is no such accepted standard for audio. After investigating several alternatives, we originally settled on the use of the AudioFile system for managing networked audio resources in TRIPS. AudioFile was developed by DEC and is based on the design (and much of the code) of the X Window System.

Just as Java promises a portable graphical environment through its graphical interface classes, about halfway through the project, the first release of the Java Sound API was released, promising equally portable access to the audio capabilities of a platform.. At that point, version 0.86, it was poorly documented, somewhat strangely designed, and barely useable, although we spent some time trying to get it working to provide platform-independent audio for use in TRIPS. Working on both a Windows NT machine and our Suns, we implemented remote object versions of Java Sound objects (using Java RMI) to transfer audio across a network (essential to enable remote access). Then there was a major rewrite of the package early this year, which we switched to in the hopes that it would improve the functionality.

Now it seems that the Java Sound API is being rolled into the Java Media Framework (JMF), which is being touted as the solution to cross-platform use of multimedia resources such as audio and video. This standard is still in its infancy (for example, the 1.0 version supports only playing audio files on client machines; the 2.0 version is only supported as part of the JDK1.3). This is therefore a moving target, and in fact some needed functionality is simply not yet implemented (such as rate conversion, which we did some work on ourselves in an effort to compensate). We are confident, however, that once the standards settle down we can use the expertise we developed while working with the earlier versions to enable remote access to a platform's audio resources in a standard way.

## 4.6 Portable Access to TRIPS via Tablet Computer

One of the goals of the audio work described in the previous section was to support access to TRIPS by a remote user with a tablet computer connected via wireless network. The platform we were investigating was a Fujitsu Stylistic 1200 obtained as part of another project. We felt that this would make a good target and a nice demo for some of the work on remote access using Java that we were doing in this project.

Unfortunately, the tablet was extremely difficult to use. First, even installing and configuring Windows NT was difficult. The lack of any fixed drive (floppy, cdrom, whatever) made life much harder. We were told by the vendor that if the drive in the machine died or we needed to reinstall Windows for some reason, that it would have to be sent back to the factory! Working carefully, we managed to get the audio capabilities of the tablet turned on as far as Windows was concerned (that is, Windows could play and record sounds). The Fujitsu appeared to use some kind of SoundBlaster-compatible audio system, although we couldn't be sure exactly what was going on. There was no documentation, of course.

Next, in order to make the tablet a viable interface device for TRIPS, we needed to support audio input and output from the tablet. TRIPS is fundamentally about conversation, so spoken language is an essential aspect of the interaction between the human and the system. As noted above, it is unlikely that we could (or would want to) run the speech recognition or speech synthesis systems on the tablet. So, as described in the previous section, we planned to use the network to ship the audio to and from another workstation using remote versions of Java Sound objects connected to the speech recognition and synthesis engines on the remote machine. This would require some integration between those engines, written in C (and a commercial, non-source program in the case of TrueTalk) and the Java Sound objects, but we felt we could handle that.

Unfortunately, it turned out that the Fujitsu tablet was simply not up to the job. It was already underpowered (Pentium I), low on memory (requiring proprietary modules to upgrade), and had a small screen (640x480). Still, it might have worked as proof of concept. But the Java Sound package simply didn't work properly, so far as we could tell, with the audio hardware in the tablet. We spent many long nights trying to make it work, but to no avail. Perhaps with newer versions of the Java Sound or JMF classes it would work, but on the other hand, those classes require the even greater overhead of newer version of the JDK, which would likely overwhelm the Fujitsu. A newer version of the tablet has been released (the Stylistic 2300). If we were to get our hands on one of those, I think we could make a very effective version of TRIPS that used the Java versions of the interface components and Java Sound/JMF for audio to enable truly portable remote access to TRIPS.

# 5 Future Work

The results of our work on using Java to provide remote access to TRIPS have been very successful, but some work remains to be done. In this section we touch briefly on some of the more interesting issues.

- Our work with applets and the TripsApplet class described in Appendix B provide a solid foundation for developing dual-purpose (applet/application) components for TRIPS. We need to complete the implementation of the "Java Desktop" to support simple and effective use of these applets from web browsers. In effect, we need to build a "Java Window Manager" that can manage the various windows put up by our applets, and provide control of and coordination between them.
- Several interface components are not yet ported to Java. These include in particular the Map Viewer and Plan Construction Window. Given the work described in this report, there are no major technical reasons why this can't be completed. However, these are fairly large, fairly complex programs, and even a fairly straightforward port will take time.
- We are continuing our work on using the Java Media Framework to provide audio support for TRIPS components. As noted above, this is unfortunately a moving target. Two separate issues need to be addressed:
  - First, we need to be able to use the JMF's platform-independent support of audio input and output to connect our existing speech recognition and speech synthesis components to the audio resources of a remote platform.
  - Second, we are investigating using the Java Speech API (part of the JMF) to connect our TRIPS components to the speech recognition and synthesis engines themselves. This would allow us to plug in new speech engines, perhaps remote ones, for example, on platforms on which our current engines are not available. We have done fairly extensive work on development of JSAPI classes (which are not part of the JavaSoft distribution). We have connected these to our TrueTalk speech synthesizer and will also connect them to our Sphinx-II speech recognizer (via JNI, the Java Native Interface). This is a complicated process, but the results would enable much easier remote access to TRIPS if we could interface directly to COTS speech engines on a remote platform.
- Finally, we would like to revisit the issue of supporting a remote TRIPS user on a portable platform. As noted above, our opinion of the Fujitsu 1200 was quite negative. However, there are newer machines than that available now, and wireless networking has also become much simpler and more affordable. The idea behind TRIPS, namely that the user is carrying on a conversation with the user, should be applicable in a wide range of situations where the user is connected to the system in more or less powerful ways (for example, from cell phone to high-powered laptop).

## 6 Conclusions

We have made good progress towards the goal of enabling remote access to TRIPS by adopting the Java platform for interface development. When we started, we weren't even sure that Java was a practical alternative for system development. This project has enabled us to experiment with and validate a large number of Java technologies, including:

1. Basic language features
2. User interface components, design, and functionality
3. Networking and other capabilities for KQML message-passing interaction with the rest of TRIPS via the TRIPS Input Manager (facilitator)
4. Reimplementation of existing TRIPS components in Java, and development of infrastructure to support future component development
5. The Java security model and the applet signing process for privileged execution of applets on remote hosts
6. Access to audio resources in a platform-independent manner via the Java Sound classes
7. Remote access to Java objects via the Java Remote Method Invocation (RMI) facilities
8. Integration of legacy and COTS programs with Java components via the Java Native Interface (JNI) specification
9. Provision and use of speech recognition and synthesis capabilities through an implementation of the Java Speech API connected to the Sphinx-II recognizer and TrueTalk synthesizer (both commercial products)
10. Use of Java on a portable tablet computer connected via wireless LAN
11. Changes to TRIPS to support a user interacting via the tablet computer

Clearly Java was the right choice for the future, and the support under this project was crucial to the development of the next generation of TRIPS.

## 7 References


[Allen, et al., 1995] James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, and David R. Traum, "The TRAINS Project: A case study in defining a conversational planning agent," *Journal of Experimental and Theoretical AI*, 7:7-48, 1995.

[Ferguson and Allen, 1998] George Ferguson and James F. Allen, "TRIPS: An Integrated Intelligent Problem-Solving Assistant," In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 27-30 July 1998.

[Ferguson, Allen, and Miller, 1996] George Ferguson, James Allen, and Brad Miller, "TRAINS-95: Towards a Mixed-Initiative Planning Assistant," In

Brian Drabble, editor, *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70--77, Edinburgh, Scotland, 29-31 May 1996.

[Ferguson et al., 1996] George Ferguson, James F. Allen, Brad W. Miller, and Eric K. Ringger, "The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant," TRAINS Technical Note 96-5, Department of Computer Science, University of Rochester, Rochester, NY, October 1996.

[Finin et al., 1993] Tim Finin, Jay Weber, Gio Wiederhold, Michael Genesereth, Richard Fritzson, Donald McKay, James McGuire, Richard Pelavin, Stuart Shapiro, and Chris Beck, "Specification of the KQML Agent-Communication Language," draft, 15 June 1993.

# A. Documentation for Package TRIPS.KQML

## Interface Summary

| | |
|---|---|
| *KQMLReceiver* | |

## Class Summary

| | |
|---|---|
| **KQMLList** | Class representing KQML lists. |
| **KQMLObject** | Base class for all KQML objects (KQMLPerformative, KQMLList, etc.). |
| **KQMLPerformative** | A class representing KQML performatives. |
| **KQMLQuotation** | A class representation quotations in KQML. |
| **KQMLReader** | A class for reading KQML performatives from an InputStream. |
| **KQMLReaderThread** | |
| **KQMLString** | A class representing KQML strings. |

## Exception Summary

| | |
|---|---|
| KQMLBadCharacterException | Thrown when a non-KQML character is read. |
| KQMLBadCloseException | Thrown when a closing parenthesis was expected but not read. |
| KQMLBadCommaException | Thrown when a comma is read outside of a backquoted expression. |
| KQMLBadHashException | Thrown when an illegal ``hashed string'' syntax is detected (it should be ``#"'. |
| KQMLBadOpenException | Thrown when an open parenthesis was read when one was not expected. |
| KQMLBadPerformativeException | Thrown when the expression read is not a performative (or actually, not a list, since we don't check that it's actually a verb followed by keyword/value pairs). |
| KQMLException | Parent class of all exceptions thrown during KQML I/O. |
| KQMLExpectedWhitespaceException | Thrown when whitespace is expected but something else is read. |

## A.1. Interface KQMLReceiver

public abstract interface **KQMLReceiver**

## Method Summary

| | |
|---|---|
| void | **receiveAchieve**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAdvertise**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskIf**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskOne**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveBroadcast**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveBrokerAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveBrokerOne**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeleteAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeleteOne**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeny**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDiscard**(KQMLPerformative msg) |
| void | **receiveEOF**() |
| void | **receiveEos**(KQMLPerformative msg) |
| void | **receiveError**(KQMLPerformative msg) |
| void | **receiveForward**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveInsert**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveMessageMissingContent**(KQMLPerformative msg) |
| void | **receiveMessageMissingVerb**(KQMLPerformative msg) |
| void | **receiveNext**(KQMLPerformative msg) |

| | |
|---|---|
| void | **receiveOtherPerformative**(KQMLPerformative msg) |
| void | **receiveReady**(KQMLPerformative msg) |
| void | **receiveRecommendAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecommendOne**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecruitAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecruitOne**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRegister**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveReply**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRequest**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRest**(KQMLPerformative msg) |
| void | **receiveSorry**(KQMLPerformative msg) |
| void | **receiveStandby**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveStreamAll**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveSubscribe**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveTell**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveTransportAddress**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnachieve**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnadvertise**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUndelete**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUninsert**(KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnregister**(KQMLPerformative msg) |
| void | **receiveUntell**(KQMLPerformative msg, java.lang.Object content) |

# Method Detail

## receiveEOF

```
public void receiveEOF()
```

---

## receiveMessageMissingVerb

```
public void receiveMessageMissingVerb(KQMLPerformative msg)
```

---

## receiveMessageMissingContent

```
public void receiveMessageMissingContent(KQMLPerformative msg)
```

---

## receiveAskIf

```
public void receiveAskIf(KQMLPerformative msg,
                         java.lang.Object content)
```

---

## receiveAskAll

```
public void receiveAskAll(KQMLPerformative msg,
                          java.lang.Object content)
```

---

## receiveAskOne

```
public void receiveAskOne(KQMLPerformative msg,
                          java.lang.Object content)
```

---

## receiveStreamAll

```
public void receiveStreamAll(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveTell

```
public void receiveTell(KQMLPerformative msg,
                        java.lang.Object content)
```

---

## receiveUntell

```
public void receiveUntell(KQMLPerformative msg,
                          java.lang.Object content)
```

---

## receiveDeny

```
public void receiveDeny(KQMLPerformative msg,
                        java.lang.Object content)
```

---

## receiveInsert

```
public void receiveInsert(KQMLPerformative msg,
                          java.lang.Object content)
```

---

## receiveUninsert

```
public void receiveUninsert(KQMLPerformative msg,
                            java.lang.Object content)
```

---

## receiveDeleteOne

```
public void receiveDeleteOne(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveDeleteAll

```
public void receiveDeleteAll(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveUndelete

```
public void receiveUndelete(KQMLPerformative msg,
                            java.lang.Object content)
```

---

## receiveAchieve

```
public void receiveAchieve(KQMLPerformative msg,
                           java.lang.Object content)
```

---

## receiveUnachieve

```
public void receiveUnachieve(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveAdvertise

```
public void receiveAdvertise(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveUnadvertise

```
public void receiveUnadvertise(KQMLPerformative msg,
                               java.lang.Object content)
```

---

## receiveSubscribe

```
public void receiveSubscribe(KQMLPerformative msg,
```

```
                                java.lang.Object content)
```

### receiveStandby

```
public void receiveStandby(KQMLPerformative msg,
                           java.lang.Object content)
```

### receiveRegister

```
public void receiveRegister(KQMLPerformative msg,
                            java.lang.Object content)
```

### receiveForward

```
public void receiveForward(KQMLPerformative msg,
                           java.lang.Object content)
```

### receiveBroadcast

```
public void receiveBroadcast(KQMLPerformative msg,
                             java.lang.Object content)
```

### receiveTransportAddress

```
public void receiveTransportAddress(KQMLPerformative msg,
                                    java.lang.Object content)
```

### receiveBrokerOne

```
public void receiveBrokerOne(KQMLPerformative msg,
                             java.lang.Object content)
```

## receiveBrokerAll

```
public void receiveBrokerAll(KQMLPerformative msg,
                             java.lang.Object content)
```

---

## receiveRecommendOne

```
public void receiveRecommendOne(KQMLPerformative msg,
                                java.lang.Object content)
```

---

## receiveRecommendAll

```
public void receiveRecommendAll(KQMLPerformative msg,
                                java.lang.Object content)
```

---

## receiveRecruitOne

```
public void receiveRecruitOne(KQMLPerformative msg,
                              java.lang.Object content)
```

---

## receiveRecruitAll

```
public void receiveRecruitAll(KQMLPerformative msg,
                              java.lang.Object content)
```

---

## receiveReply

```
public void receiveReply(KQMLPerformative msg,
                         java.lang.Object content)
```

---

## receiveRequest

```
public void receiveRequest(KQMLPerformative msg,
                           java.lang.Object content)
```

---

### receiveEos

`public void receiveEos(KQMLPerformative msg)`

---

### receiveError

`public void receiveError(KQMLPerformative msg)`

---

### receiveSorry

`public void receiveSorry(KQMLPerformative msg)`

---

### receiveReady

`public void receiveReady(KQMLPerformative msg)`

---

### receiveNext

`public void receiveNext(KQMLPerformative msg)`

---

### receiveRest

`public void receiveRest(KQMLPerformative msg)`

---

### receiveDiscard

`public void receiveDiscard(KQMLPerformative msg)`

---

## receiveUnregister

```
public void receiveUnregister(KQMLPerformative msg)
```

---

## receiveOtherPerformative

```
public void receiveOtherPerformative(KQMLPerformative msg)
```

---

## A.2. Class KQMLList

```
java.lang.Object
  |
  +--TRIPS.KQML.KQMLObject
        |
        +--TRIPS.KQML.KQMLList
```

public class **KQMLList**

extends KQMLObject

Class representing KQML lists. These are really just Vectors that print nicely using KQML syntax.

**See Also:**

KQMLReader

| Constructor Summary |
|---|
| **KQMLList**() <br> Returns a new empty KQMLList. |
| **KQMLList**(java.lang.Object a1) |
| **KQMLList**(java.lang.Object a1, java.lang.Object a2) |
| **KQMLList**(java.lang.Object a1, java.lang.Object a2, java.lang.Object a3) |
| **KQMLList**(java.lang.Object a1, java.lang.Object a2, java.lang.Object a3, java.lang.Object a4) |
| **KQMLList**(java.lang.Object a1, java.lang.Object a2, java.lang.Object a3, java.lang.Object a4, java.lang.Object a5) |

| Method Summary | |
|---|---|
| void | **add**(java.lang.Object obj) <br> Adds an element to the end of a KQMLList. |
| java.lang.Object | **getKeywordArg**(java.lang.String keyword) <br> Returns the object following the given keyword in the list. |
| void | **insertAt**(java.lang.Object obj, int index) <br> Inserts an element at the given index of an KQMLList. |
| int | **length**() <br> Returns the length of a KQMLList. |
| java.lang.Object | **nth**(int n) <br> Returns the requested element of a KQMLList. |

| | |
|---|---|
| void | **push**(java.lang.Object obj)<br>Adds an element to the front a KQMLList. |
| void | **removeAt**(int index)<br>Removes the element at the given index of an KQMLList. |
| java.lang.String | **toString**()<br>Returns a KQMLList as a String in KQML syntax. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

## KQMLList

```
public KQMLList()
```

Returns a new empty KQMLList.

**See Also:**

KQMLPerformative, KQMLReader

---

## KQMLList

```
public KQMLList(java.lang.Object a1)
```

---

## KQMLList

```
public KQMLList(java.lang.Object a1,
                java.lang.Object a2)
```

---

## KQMLList

```
public KQMLList(java.lang.Object a1,
                java.lang.Object a2,
                java.lang.Object a3)
```

---

## KQMLList

```
public KQMLList(java.lang.Object a1,
                java.lang.Object a2,
                java.lang.Object a3,
                java.lang.Object a4)
```

---

### KQMLList

```
public KQMLList(java.lang.Object a1,
                java.lang.Object a2,
                java.lang.Object a3,
                java.lang.Object a4,
                java.lang.Object a5)
```

## Method Detail

### add

```
public void add(java.lang.Object obj)
```

Adds an element to the end of a KQMLList.

**Parameters:**

`obj` - Object to add

---

### push

```
public void push(java.lang.Object obj)
```

Adds an element to the front a KQMLList.

**Parameters:**

`obj` - Object to add

---

### insertAt

```
public void insertAt(java.lang.Object obj,
                     int index)
```

Inserts an element at the given index of an KQMLList.

**Parameters:**

`obj` - Object to add

`index` - Index at which to insert

---

## removeAt

`public void removeAt(int index)`

Removes the element at the given index of an KQMLList.

**Parameters:**

`index` - Index at which to delete

---

## nth

`public java.lang.Object nth(int n)`

Returns the requested element of a KQMLList.

**Parameters:**

`n` - Index of object

**Returns:**

Object at that index

---

## length

`public int length()`

Returns the length of a KQMLList.

**Returns:**

Length of list

---

## getKeywordArg

`public java.lang.Object getKeywordArg(java.lang.String keyword)`

Returns the object following the given keyword in the list. Uses case-insensitive matching on the keyword.

**Parameters:**

`keyword` - Name of parameter (including colon)

**Returns:**

Value of parameter (String, KQMLString, KQMLQuotation, or KQMLList)

**See Also:**

`String`, `KQMLString`, `KQMLQuotation`, `KQMLList`

---

## toString

`public java.lang.String **toString**()`

Returns a KQMLList as a String in KQML syntax.

**Returns:**

String denoting KQMLList

**Overrides:**

toString in class java.lang.Object

---

## A.3. Class KQMLObject

```
java.lang.Object
   |
   +--TRIPS.KQML.KQMLObject
```

**Direct Known Subclasses:**

KQMLList, KQMLPerformative, KQMLQuotation, KQMLString

---

public class **KQMLObject**

extends java.lang.Object

Base class for all KQML objects (KQMLPerformative, KQMLList, etc.).

**See Also:**

KQMLReader

---

| Constructor Summary | |
|---|---|
| `KQMLObject()` | |

| Methods inherited from class java.lang.Object |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait` |

## Constructor Detail

### KQMLObject

`public KQMLObject()`

---

## A.4. Class KQMLPerformative

```
java.lang.Object
  |
  +--TRIPS.KQML.KQMLObject
        |
        +--TRIPS.KQML.KQMLPerformative
```

---

public class **KQMLPerformative**

extends KQMLObject

A class representing KQML performatives. This is really just a Vector with methods for getting at the verb and parameters of the performative.

**See Also:**

KQMLReader

---

| Constructor Summary |
|---|
| **KQMLPerformative**(KQMLList list)<br>Creates a new performative from the given list. |
| **KQMLPerformative**(java.lang.String verb)<br>Creates a new performative with the given verb (and no parameters). |

| Method Summary | |
|---|---|
| java.lang.Object | **getParameter**(java.lang.String keyword)<br>Returns the requested parameter of the performative. |
| java.lang.String | **getVerb**()<br>Returns the verb of the performative as a String. |
| void | **setParameter**(java.lang.String keyword, java.lang.Object value)<br>Sets the given parameter of the performative. |
| java.lang.String | **toString**()<br>Returns the performative as a String. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## Constructor Detail

### KQMLPerformative

`public KQMLPerformative(java.lang.String verb)`

Creates a new performative with the given verb (and no parameters).

**Parameters:**

`str` - The verb of the performative

---

### KQMLPerformative

`public KQMLPerformative(KQMLList list)`

Creates a new performative from the given list. Note that this constructor does not currently check that the elements of the list are in fact a verb followed by keyword/value pairs.

**Parameters:**

`list` - KQMLList containing elements of the performative.

## Method Detail

### getVerb

`public java.lang.String getVerb()`

Returns the verb of the performative as a String.

**Returns:**

Verb of performative

---

### getParameter

`public java.lang.Object getParameter(java.lang.String keyword)`

Returns the requested parameter of the performative. The case of the given keyword is ignored.

**Parameters:**

`keyword` - Name of parameter (including colon)

**Returns:**

Value of parameter (String, KQMLString, KQMLQuotation, or KQMLList)

**See Also:**

String, KQMLString, KQMLQuotation, KQMLList

---

## setParameter

```
public void setParameter(java.lang.String keyword,
                         java.lang.Object value)
```

Sets the given parameter of the performative.

**Parameters:**

keyword - Name of parameter (including colon)

value - Value of parameter (String, KQMLString, KQMLQuotation, or KQMLList)

**See Also:**

String, KQMLString, KQMLQuotation, KQMLList

---

## toString

```
public java.lang.String toString()
```

Returns the performative as a String.

**Returns:**

String suitable for printing as KQML

**Overrides:**

toString in class java.lang.Object

---

## A.5. Class KQMLQuotation

```
java.lang.Object
  |
  +--TRIPS.KQML.KQMLObject
        |
        +--TRIPS.KQML.KQMLQuotation
```

---

public class **KQMLQuotation**

extends KQMLObject

A class representation quotations in KQML. These are expressions preceded by a quote, backquote, or comma (the ``type" of the quotation).

**See Also:**

KQMLPerformative, KQMLReader

---

| Constructor Summary |
|---|
| **KQMLQuotation**(char t, java.lang.Object obj)<br>Returns a new KQMLQuotation consisting of the given elemnts. |

| Method Summary | |
|---|---|
| java.lang.Object | **getObject**()<br>Returns the object being quoted. |
| char | **getType**()<br>Returns the type of the quotation. |
| java.lang.String | **toString**()<br>Returns a KQMLQuotation as a String in KQML syntax. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Constructor Detail

### KQMLQuotation

```
public KQMLQuotation(char t,
                     java.lang.Object obj)
```

Returns a new KQMLQuotation consisting of the given elemnts.

**Parameters:**

`t` - Type of quotation (quote, backquote, or comma)

`obj` - Object being quoted (String, KQMLString, KQMLQuotation, or KQMLList)

# Method Detail

## getType

`public char getType()`

Returns the type of the quotation.

**Returns:**

Type of quotation (quote, backquote, or comma)

---

## getObject

`public java.lang.Object getObject()`

Returns the object being quoted.

**Returns:**

Object being quoted (String, KQMLString, KQMLQuotation, or KQMLList)

---

## toString

`public java.lang.String toString()`

Returns a KQMLQuotation as a String in KQML syntax.

**Returns:**

String denoting KQMLQuotation

**Overrides:**

toString in class java.lang.Object

---

## A.6. Class KQMLReader

```
java.lang.Object
  |
  +--java.io.Reader
        |
        +--java.io.InputStreamReader
              |
              +--TRIPS.KQML.KQMLReader
```

---

public class **KQMLReader**

extends java.io.InputStreamReader

A class for reading KQML performatives from an InputStream. For example:

```
KQMLReader in = new KQMLReader(socket.getInputStream());
```

**See Also:**

KQMLPerformative

---

| Fields inherited from class java.io.Reader |
| --- |
| lock |

| Constructor Summary | |
| --- | --- |
| **KQMLReader**(java.io.InputStream s)<br>Creates a new stream from which to read KQML Performatives. | |

| Method Summary | |
| --- | --- |
| static void | **main**(java.lang.String[] a)<br>For testing. |
| int | **read**() |
| KQMLPerformative | **readPerformative**()<br>Reads a performative. |

| Methods inherited from class java.io.InputStreamReader |
| --- |
| close, getEncoding, read, ready |

| Methods inherited from class java.io.Reader |
| --- |
| mark, markSupported, read, reset, skip |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, |

```
wait, wait, wait
```

# Constructor Detail

## KQMLReader

```
public KQMLReader(java.io.InputStream s)
```

Creates a new stream from which to read KQML Performatives.

**Parameters:**

`s` - InputStream from which to read

**See Also:**

`InputStream`

# Method Detail

## read

```
public int read()
         throws java.io.IOException
```

**Overrides:**

read in class java.io.InputStreamReader

---

## readPerformative

```
public KQMLPerformative readPerformative()
                                  throws java.io.IOException
```

Reads a performative.

**Returns:**

Next performative from input stream

**Throws:**

KQMLException - If the input is not proper KQML.

java.io.EOFException - If EOF is reached.

java.io.IOException - For any other I/O error.

## main

```
public static void main(java.lang.String[] a)
```

For testing.

## A.7. Class KQMLReaderThread

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--TRIPS.KQML.KQMLReaderThread
```

---

public class **KQMLReaderThread**

extends java.lang.Thread

---

# Field Summary

| | |
|---|---|
| protected KQMLReader | **reader** |
| protected KQMLReceiver | **receiver** |
| protected boolean | **stopped** |
| protected boolean | **suspended** |

| Fields inherited from class java.lang.Thread |
|---|
| MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY |

# Constructor Summary

| | |
|---|---|
| **KQMLReaderThread**(KQMLReceiver rec, KQMLReader in) | |

# Method Summary

| | |
|---|---|
| protected void | **receive**(KQMLPerformative msg) |
| void | **run**() |
| void | **stopSafely**() |
| void | **suspendSafely**() |

| Methods inherited from class java.lang.Thread |
|---|

activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

# Field Detail

### receiver

`protected KQMLReceiver **receiver**`

---

### reader

`protected KQMLReader **reader**`

---

### stopped

`protected volatile boolean **stopped**`

---

### suspended

`protected volatile boolean **suspended**`

# Constructor Detail

### KQMLReaderThread

`public **KQMLReaderThread**(KQMLReceiver rec,` KQMLReader in)

# Method Detail

### run

```
public void run()
```

**Overrides:**

run in class java.lang.Thread

### stopSafely

```
public void stopSafely()
```

### suspendSafely

```
public void suspendSafely()
```

### receive

```
protected void receive(KQMLPerformative msg)
```

## A.8. Class KQMLString

```
java.lang.Object
  |
  +--TRIPS.KQML.KQMLObject
        |
        +--TRIPS.KQML.KQMLString
```

---

public class **KQMLString**

extends KQMLObject

A class representing KQML strings. These are just regular strings that print themselves using KQML syntax.

**See Also:**

KQMLPerformative, KQMLReader

---

| Constructor Summary | |
|---|---|
| **KQMLString**() <br> Creates a new empty KQMLString. | |
| **KQMLString**(java.lang.String s) <br> Creates a new KQMLString with the given contents. | |

| Method Summary | |
|---|---|
| int | **charAt**(int n) <br> Returns the character at a given index in a KQMLString. |
| int | **length**() <br> Returns the number of characters in a KQMLString. |
| java.lang.String | **stringValue**() <br> Returns the String content of a KQMLString (no extra quotes). |
| java.lang.String | **toString**() <br> Returns a KQMLString as a String in KQML syntax. |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## Constructor Detail

### KQMLString

`public KQMLString()`

Creates a new empty KQMLString.

---

### KQMLString

`public KQMLString(java.lang.String s)`

Creates a new KQMLString with the given contents.

**Parameters:**

`s` - Contents of string.

## Method Detail

### length

`public int length()`

Returns the number of characters in a KQMLString.

**Returns:**

Length of KQMLString

---

### charAt

`public int charAt(int n)`

Returns the character at a given index in a KQMLString.

**Parameters:**

`n` - Index of character

**Returns:**

Character at that index

---

### toString

`public java.lang.String toString()`

Returns a KQMLString as a String in KQML syntax.

**Returns:**

String denoting KQMLString

**Overrides:**

toString in class java.lang.Object

---

## stringValue

```
public java.lang.String stringValue()
```

Returns the String content of a KQMLString (no extra quotes).

**Returns:**

String contents of KQMLString

---

## A.9. Class KQMLBadCharacterException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadCharacterException
```

---

public class **KQMLBadCharacterException**

extends KQMLException

Thrown when a non-KQML character is read.

**See Also:**

KQMLReader, Serialized Form

---

| Method Summary | |
|---|---|
| java.lang.String | **toString**() |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Method Detail

### toString

`public java.lang.String toString()`

**Overrides:**

toString in class java.lang.Throwable

## A.10. Class KQMLBadCloseException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadCloseException
```

---

public class **KQMLBadCloseException**

extends KQMLException

Thrown when a closing parenthesis was expected but not read. (In fact, this should never be thrown, but...)

**See Also:**

KQMLReader, Serialized Form

---

# Method Summary

| | |
|---|---|
| java.lang.String | **toString**() |

### Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

# Method Detail

### toString

public java.lang.String **toString**()

**Overrides:**

toString in class java.lang.Throwable

## A.11. Class KQMLBadCommaException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadCommaException
```

public class **KQMLBadCommaException**

extends KQMLException

Thrown when a comma is read outside of a backquoted expression.

**See Also:**

KQMLReader, Serialized Form

# Method Summary

| | |
|---|---|
| java.lang.String | **toString**() |

### Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

# Method Detail

## toString

```
public java.lang.String toString()
```

**Overrides:**

toString in class java.lang.Throwable

## A.12. Class KQMLBadHashException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadHashException
```

---

public class **KQMLBadHashException**

extends KQMLException

Thrown when an illegal ``hashed string" syntax is detected (it should be ``#"". This is usually caused by a hash (pound) character getting into the input by accident, since hashed strings are rarely used. They can be printed by Lisp, for example, when printing structures without print functions.

**See Also:**

KQMLReader, Serialized Form

---

| Method Summary | |
|---|---|
| java.lang.String | **toString**() |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Method Detail

### toString

`public java.lang.String toString()`

**Overrides:**

toString in class java.lang.Throwable

## A.13. Class KQMLBadOpenException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadOpenException
```

---

public class **KQMLBadOpenException**

extends KQMLException

Thrown when an open parenthesis was read when one was not expected. (In fact, this should never be thrown...)

**See Also:**

KQMLReader, Serialized Form

---

| Method Summary | |
|---|---|
| java.lang.String | **toString**() |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Method Detail

### toString

`public java.lang.String toString()`

**Overrides:**

toString in class java.lang.Throwable

## A.14. Class KQMLBadPerformativeException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLBadPerformativeException
```

---

public class **KQMLBadPerformativeException**

extends KQMLException

Thrown when the expression read is not a performative (or actually, not a list, since we don't check that it's actually a verb followed by keyword/value pairs).

**See Also:**

KQMLReader, Serialized Form

---

| Method Summary | |
|---|---|
| java.lang.String | **toString**() |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

# Method Detail

### toString

`public java.lang.String toString()`

**Overrides:**

toString in class java.lang.Throwable

## A.15. Class KQMLException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
```

**Direct Known Subclasses:**

KQMLBadCharacterException, KQMLBadCloseException, KQMLBadCommaException, KQMLBadHashException, KQMLBadOpenException, KQMLBadPerformativeException, KQMLExpectedWhitespaceException

---

public class **KQMLException**

extends java.io.IOException

Parent class of all exceptions thrown during KQML I/O. This is a subclass of IOException so that applications that don't care about the details of an error can just catch them all.

**See Also:**

`KQMLReader`, Serialized Form

---

| **Methods inherited from class java.lang.Throwable** |
|---|
| `fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString` |

| **Methods inherited from class java.lang.Object** |
|---|
| `clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait` |

## A.16. Class KQMLExpectedWhitespaceException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--TRIPS.KQML.KQMLException
                          |
                          +--TRIPS.KQML.KQMLExpectedWhitespaceException
```

---

public class **KQMLExpectedWhitespaceException**

extends KQMLException

Thrown when whitespace is expected but something else is read.

**See Also:**

KQMLReader, Serialized Form

---

| Method Summary | |
|---|---|
| java.lang.String | **toString**() |

| Methods inherited from class java.lang.Throwable |
|---|
| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace |

| Methods inherited from class java.lang.Object |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## Method Detail

### toString

`public java.lang.String toString()`

**Overrides:**

toString in class java.lang.Throwable

# B. Documentation for Package TRIPS.TripsApplet

| Class Summary | |
|---|---|
| TripsApplet | |
| TripsAppletFrame | Provides a frame so an applet can be run as an application AppletStub and AppletContext are implemented to provide a minimal browserlike interface to avoid crashes from browser applet specific calls like showStatus(). |
| TripsAppletFrameCloser | |

## B.1. Class TripsApplet

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--TRIPS.TripsApplet.TripsApplet
```

---

public class **TripsApplet**

extends java.applet.Applet

implements TRIPS.KQML.KQMLReceiver

**See Also:**

Serialized Form

---

| Field Summary | |
|---|---|
| protected static java.lang.String | **DEFAULT_HOST** |
| protected static int | **DEFAULT_PORT** |
| protected java.lang.String | **groupName** |
| protected java.lang.String | **host** |
| protected TRIPS.KQML.KQMLReader | **in** |
| protected java.lang.String | **moduleName** |
| protected java.io.PrintWriter | **out** |
| protected java.lang.String[] | **parameters** |
| protected int | **port** |
| protected TRIPS.KQML.KQMLReaderThread | **reader** |
| protected boolean | **useStdio** |

| Fields inherited from class java.awt.Component |
|---|
| BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT |

| Constructor Summary | |
|---|---|
| **TripsApplet**() | |
| **TripsApplet**(java.lang.String[] argv) | |

| Method Summary | |
|---|---|
| protected void | **connect**(java.lang.String host, int startport) |
| protected void | **debug**(java.lang.String msg) |
| void | **destroy**() |
| protected void | **errorReply**(TRIPS.KQML.KQMLPerformative msg, java.lang.String comment) |
| void | **exit**(int n) |
| java.lang.String | **getParameter**(java.lang.String parm) |
| protected void | **handleCommonParameters**() |
| void | **init**() |
| void | **receiveAchieve**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAdvertise**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskIf**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveAskOne**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveBroadcast**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveBrokerAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |

| | |
|---|---|
| void | **receiveBrokerOne**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeleteAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeleteOne**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDeny**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveDiscard**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveEOF**() |
| void | **receiveEos**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveError**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveForward**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveInsert**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveMessageMissingContent**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveMessageMissingVerb**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveNext**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveOtherPerformative**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveReady**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveRecommendAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecommendOne**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecruitAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRecruitOne**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRegister**(TRIPS.KQML.KQMLPerformative msg, |

| | |
|---|---|
| | java.lang.Object content) |
| void | **receiveReply**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRequest**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveRest**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveSorry**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveStandby**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveStreamAll**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveSubscribe**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveTell**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveTransportAddress**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnachieve**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnadvertise**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUndelete**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUninsert**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| void | **receiveUnregister**(TRIPS.KQML.KQMLPerformative msg) |
| void | **receiveUntell**(TRIPS.KQML.KQMLPerformative msg, java.lang.Object content) |
| protected void | **register**() |
| protected void | **send**(TRIPS.KQML.KQMLPerformative msg) |
| protected void | **sendReadyMessage**() |

| | |
|---|---|
| void | **start**() |
| void | **stop**() |
| protected void | **warn**(java.lang.String msg) |

## Methods inherited from class java.applet.Applet

getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus

## Methods inherited from class java.awt.Panel

addNotify

## Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFont, setLayout, update, validate, validateTree

## Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getInputContext, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus

#### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Field Detail

### parameters

`protected java.lang.String[] parameters`

---

### host

`protected java.lang.String host`

---

### port

`protected int port`

---

### useStdio

`protected boolean useStdio`

---

### moduleName

`protected java.lang.String moduleName`

---

### groupName

`protected java.lang.String groupName`

---

### reader

`protected TRIPS.KQML.KQMLReaderThread reader`

---

### in

`protected TRIPS.KQML.KQMLReader in`

---

### out

`protected java.io.PrintWriter out`

---

### DEFAULT_HOST

`protected static java.lang.String DEFAULT_HOST`

---

### DEFAULT_PORT

`protected static int DEFAULT_PORT`

## Constructor Detail

### TripsApplet

`public TripsApplet(java.lang.String[] argv)`

---

### TripsApplet

`public TripsApplet()`

## Method Detail

### init

`public void init()`

**Overrides:**

init in class java.applet.Applet

---

### start

```
public void start()
```

**Overrides:**

start in class java.applet.Applet

---

### stop

```
public void stop()
```

**Overrides:**

stop in class java.applet.Applet

---

### destroy

```
public void destroy()
```

**Overrides:**

destroy in class java.applet.Applet

---

### exit

```
public void exit(int n)
```

---

### handleCommonParameters

```
protected void handleCommonParameters()
```

---

### connect

```
protected void connect(java.lang.String host,
                       int startport)
```

---

### register

```
protected void register()
```

### sendReadyMessage

```
protected void sendReadyMessage()
```

### getParameter

```
public java.lang.String getParameter(java.lang.String parm)
```

**Overrides:**

getParameter in class java.applet.Applet

### receiveEOF

```
public void receiveEOF()
```

**Specified by:**

receiveEOF in interface TRIPS.KQML.KQMLReceiver

### receiveMessageMissingVerb

```
public void receiveMessageMissingVerb(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveMessageMissingVerb in interface TRIPS.KQML.KQMLReceiver

### receiveMessageMissingContent

```
public void
   receiveMessageMissingContent(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveMessageMissingContent in interface TRIPS.KQML.KQMLReceiver

### receiveAskIf

```
public void receiveAskIf(TRIPS.KQML.KQMLPerformative msg,
                         java.lang.Object content)
```

**Specified by:**

receiveAskIf in interface TRIPS.KQML.KQMLReceiver

---

## receiveAskAll

```
public void receiveAskAll(TRIPS.KQML.KQMLPerformative msg,
                          java.lang.Object content)
```

**Specified by:**

receiveAskAll in interface TRIPS.KQML.KQMLReceiver

---

## receiveAskOne

```
public void receiveAskOne(TRIPS.KQML.KQMLPerformative msg,
                          java.lang.Object content)
```

**Specified by:**

receiveAskOne in interface TRIPS.KQML.KQMLReceiver

---

## receiveStreamAll

```
public void receiveStreamAll(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveStreamAll in interface TRIPS.KQML.KQMLReceiver

---

## receiveTell

```
public void receiveTell(TRIPS.KQML.KQMLPerformative msg,
                        java.lang.Object content)
```

**Specified by:**

receiveTell in interface TRIPS.KQML.KQMLReceiver

---

### receiveUntell

```
public void receiveUntell(TRIPS.KQML.KQMLPerformative msg,
                          java.lang.Object content)
```

**Specified by:**

receiveUntell in interface TRIPS.KQML.KQMLReceiver

---

### receiveDeny

```
public void receiveDeny(TRIPS.KQML.KQMLPerformative msg,
                        java.lang.Object content)
```

**Specified by:**

receiveDeny in interface TRIPS.KQML.KQMLReceiver

---

### receiveInsert

```
public void receiveInsert(TRIPS.KQML.KQMLPerformative msg,
                          java.lang.Object content)
```

**Specified by:**

receiveInsert in interface TRIPS.KQML.KQMLReceiver

---

### receiveUninsert

```
public void receiveUninsert(TRIPS.KQML.KQMLPerformative msg,
                            java.lang.Object content)
```

**Specified by:**

receiveUninsert in interface TRIPS.KQML.KQMLReceiver

---

### receiveDeleteOne

```
public void receiveDeleteOne(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveDeleteOne in interface TRIPS.KQML.KQMLReceiver

### receiveDeleteAll

```
public void receiveDeleteAll(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveDeleteAll in interface TRIPS.KQML.KQMLReceiver

---

### receiveUndelete

```
public void receiveUndelete(TRIPS.KQML.KQMLPerformative msg,
                            java.lang.Object content)
```

**Specified by:**

receiveUndelete in interface TRIPS.KQML.KQMLReceiver

---

### receiveAchieve

```
public void receiveAchieve(TRIPS.KQML.KQMLPerformative msg,
                           java.lang.Object content)
```

**Specified by:**

receiveAchieve in interface TRIPS.KQML.KQMLReceiver

---

### receiveUnachieve

```
public void receiveUnachieve(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveUnachieve in interface TRIPS.KQML.KQMLReceiver

---

### receiveAdvertise

```
public void receiveAdvertise(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveAdvertise in interface TRIPS.KQML.KQMLReceiver

## receiveUnadvertise

```
public void receiveUnadvertise(TRIPS.KQML.KQMLPerformative msg,
                               java.lang.Object content)
```

**Specified by:**

receiveUnadvertise in interface TRIPS.KQML.KQMLReceiver

## receiveSubscribe

```
public void receiveSubscribe(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveSubscribe in interface TRIPS.KQML.KQMLReceiver

## receiveStandby

```
public void receiveStandby(TRIPS.KQML.KQMLPerformative msg,
                           java.lang.Object content)
```

**Specified by:**

receiveStandby in interface TRIPS.KQML.KQMLReceiver

## receiveRegister

```
public void receiveRegister(TRIPS.KQML.KQMLPerformative msg,
                            java.lang.Object content)
```

**Specified by:**

receiveRegister in interface TRIPS.KQML.KQMLReceiver

## receiveForward

```
public void receiveForward(TRIPS.KQML.KQMLPerformative msg,
                           java.lang.Object content)
```

**Specified by:**

receiveForward in interface TRIPS.KQML.KQMLReceiver

---

## receiveBroadcast

```
public void receiveBroadcast(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveBroadcast in interface TRIPS.KQML.KQMLReceiver

---

## receiveTransportAddress

```
public void receiveTransportAddress(TRIPS.KQML.KQMLPerformative msg,
                                    java.lang.Object content)
```

**Specified by:**

receiveTransportAddress in interface TRIPS.KQML.KQMLReceiver

---

## receiveBrokerOne

```
public void receiveBrokerOne(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveBrokerOne in interface TRIPS.KQML.KQMLReceiver

---

## receiveBrokerAll

```
public void receiveBrokerAll(TRIPS.KQML.KQMLPerformative msg,
                             java.lang.Object content)
```

**Specified by:**

receiveBrokerAll in interface TRIPS.KQML.KQMLReceiver

---

## receiveRecommendOne

```
public void receiveRecommendOne(TRIPS.KQML.KQMLPerformative msg,
                                java.lang.Object content)
```

**Specified by:**

receiveRecommendOne in interface TRIPS.KQML.KQMLReceiver

---

## receiveRecommendAll

```
public void receiveRecommendAll(TRIPS.KQML.KQMLPerformative msg,
                                java.lang.Object content)
```

**Specified by:**

receiveRecommendAll in interface TRIPS.KQML.KQMLReceiver

---

## receiveRecruitOne

```
public void receiveRecruitOne(TRIPS.KQML.KQMLPerformative msg,
                              java.lang.Object content)
```

**Specified by:**

receiveRecruitOne in interface TRIPS.KQML.KQMLReceiver

---

## receiveRecruitAll

```
public void receiveRecruitAll(TRIPS.KQML.KQMLPerformative msg,
                              java.lang.Object content)
```

**Specified by:**

receiveRecruitAll in interface TRIPS.KQML.KQMLReceiver

---

## receiveReply

```
public void receiveReply(TRIPS.KQML.KQMLPerformative msg,
                         java.lang.Object content)
```

**Specified by:**

receiveReply in interface TRIPS.KQML.KQMLReceiver

---

### receiveRequest

```
public void receiveRequest(TRIPS.KQML.KQMLPerformative msg,
                           java.lang.Object content)
```

**Specified by:**

receiveRequest in interface TRIPS.KQML.KQMLReceiver

---

### receiveEos

```
public void receiveEos(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveEos in interface TRIPS.KQML.KQMLReceiver

---

### receiveError

```
public void receiveError(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveError in interface TRIPS.KQML.KQMLReceiver

---

### receiveSorry

```
public void receiveSorry(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveSorry in interface TRIPS.KQML.KQMLReceiver

---

### receiveReady

```
public void receiveReady(TRIPS.KQML.KQMLPerformative msg)
```

**Specified by:**

receiveReady in interface TRIPS.KQML.KQMLReceiver

---

### receiveNext

`public void receiveNext(TRIPS.KQML.KQMLPerformative msg)`

**Specified by:**

receiveNext in interface TRIPS.KQML.KQMLReceiver

---

### receiveRest

`public void receiveRest(TRIPS.KQML.KQMLPerformative msg)`

**Specified by:**

receiveRest in interface TRIPS.KQML.KQMLReceiver

---

### receiveDiscard

`public void receiveDiscard(TRIPS.KQML.KQMLPerformative msg)`

**Specified by:**

receiveDiscard in interface TRIPS.KQML.KQMLReceiver

---

### receiveUnregister

`public void receiveUnregister(TRIPS.KQML.KQMLPerformative msg)`

**Specified by:**

receiveUnregister in interface TRIPS.KQML.KQMLReceiver

---

### receiveOtherPerformative

`public void receiveOtherPerformative(TRIPS.KQML.KQMLPerformative msg)`

**Specified by:**

receiveOtherPerformative in interface TRIPS.KQML.KQMLReceiver

---

### send

```
protected void send(TRIPS.KQML.KQMLPerformative msg)
```

## errorReply

```
protected void errorReply(TRIPS.KQML.KQMLPerformative msg,
                          java.lang.String comment)
```

## warn

```
protected void warn(java.lang.String msg)
```

## debug

```
protected void debug(java.lang.String msg)
```

## B.2. Class TripsAppletFrame

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--TRIPS.TripsApplet.TripsAppletFrame
```

---

public class **TripsAppletFrame**

extends java.awt.Frame

implements java.applet.AppletStub, java.applet.AppletContext

Provides a frame so an applet can be run as an application AppletStub and AppletContext are implemented to provide a minimal browserlike interface to avoid crashes from browser applet specific calls like showStatus().

**See Also:**

`TripsApplet`, `TripsReader`, `Frame`, Serialized Form

---

| Fields inherited from class java.awt.Frame |
|---|
| `CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR` |

| Fields inherited from class java.awt.Component |
|---|
| `BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT` |

| Constructor Summary | |
|---|---|
| **`TripsAppletFrame`**`(TripsApplet a)` | |

| Method Summary | |
|---|---|
| `void` | **`appletResize`**`(int width, int height)` |

| | |
|---|---|
| java.applet.Applet | **getApplet**(java.lang.String name) |
| java.applet.AppletContext | **getAppletContext**() |
| java.util.Enumeration | **getApplets**() |
| java.applet.AudioClip | **getAudioClip**(java.net.URL url) |
| java.net.URL | **getCodeBase**() |
| java.net.URL | **getDocumentBase**() |
| java.awt.Image | **getImage**(java.net.URL url) |
| java.lang.String | **getParameter**(java.lang.String name) |
| boolean | **isActive**() |
| void | **showDocument**(java.net.URL url) |
| void | **showDocument**(java.net.URL url, java.lang.String target) |
| void | **showStatus**(java.lang.String status) |

## Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getFrames, getIconImage, getMenuBar, getState, getTitle, isResizable, paramString, remove, removeNotify, setCursor, setIconImage, setMenuBar, setResizable, setState, setTitle

## Methods inherited from class java.awt.Window

addWindowListener, applyResourceBundle, applyResourceBundle, dispose, getFocusOwner, getInputContext, getLocale, getOwnedWindows, getOwner, getToolkit, getWarningString, isShowing, pack, postEvent, processEvent, processWindowEvent, removeWindowListener, show, toBack, toFront

## Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, remove, removeAll, removeContainerListener, setFont, setLayout, update, validate, validateTree

## Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus

**Methods inherited from class java.lang.Object**

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

# Constructor Detail

## TripsAppletFrame

```
public TripsAppletFrame(TripsApplet a)
```

# Method Detail

## isActive

```
public boolean isActive()
```

**Specified by:**

isActive in interface java.applet.AppletStub

---

## getDocumentBase

```
public java.net.URL getDocumentBase()
```

**Specified by:**

getDocumentBase in interface java.applet.AppletStub

---

### getCodeBase

```
public java.net.URL getCodeBase()
```

**Specified by:**

getCodeBase in interface java.applet.AppletStub

---

### getParameter

```
public java.lang.String getParameter(java.lang.String name)
```

**Specified by:**

getParameter in interface java.applet.AppletStub

---

### getAppletContext

```
public java.applet.AppletContext getAppletContext()
```

**Specified by:**

getAppletContext in interface java.applet.AppletStub

---

### appletResize

```
public void appletResize(int width,
                         int height)
```

**Specified by:**

appletResize in interface java.applet.AppletStub

---

### getAudioClip

```
public java.applet.AudioClip getAudioClip(java.net.URL url)
```

**Specified by:**

getAudioClip in interface java.applet.AppletContext

## getImage

```
public java.awt.Image getImage(java.net.URL url)
```

**Specified by:**

getImage in interface java.applet.AppletContext

---

## getApplet

```
public java.applet.Applet getApplet(java.lang.String name)
```

**Specified by:**

getApplet in interface java.applet.AppletContext

---

## getApplets

```
public java.util.Enumeration getApplets()
```

**Specified by:**

getApplets in interface java.applet.AppletContext

---

## showDocument

```
public void showDocument(java.net.URL url)
```

**Specified by:**

showDocument in interface java.applet.AppletContext

---

## showDocument

```
public void showDocument(java.net.URL url,
                         java.lang.String target)
```

**Specified by:**

showDocument in interface java.applet.AppletContext

---

### showStatus

```
public void showStatus(java.lang.String status)
```

**Specified by:**

showStatus in interface java.applet.AppletContext

## B.3. Using TripsApplet Classes

The following code fragment shows how easy it is to use the TripsApplet classes to make a program operate as either an applet or an application.

First, write your code as an applet. Be sure to call `super.init()` in your `init()` method to let the `TripsApplet` class perform initializations. Use `getParameter()` to retrieve command-line or `PARAM` tag parameters (for applications and applets, respectively).

Then, if your applet is in class `FooApplet`, for example, you should create class `FooApplication` as below:

```
/*
 * FooApplication.java
 *
 * This simply wraps the FooApplet in an TripsAppletFrame.
 */

import TRIPS.TripsApplet.*;

public class FooApplication extends FooApplet {
    public static void main(String argv[]) {
     new TripsAppletFrame(new FooApplet(argv));
    }
}
```

That's it. To run your program as an application, you pass the application's class name (`FooApplication` in this example) to the java interpreter, followed by any options. For example:

```
% java FooApplication -geometry 100x25
```

To run as an applet, embed the appropriate code in an HTML page, make the applet class file(s) available from your web server, and visit the page with your browser. You could also use the `appletrunner` to test applets.

# C. Signing Applets in the JDK1.1 Security Model

This section describes the procedure to be followed to create a signed applet under the JDK 1.1 security model and using JDK 1.1 tools.

The main tool used here is `javakey`. To use `javakey` to sign an applet, making it trusted, one must first create a trusted signer identity in the `javakey` database. There must also be at least one certificate in the database associated with the trusted signer. We can obtain a certificate commercially from an authorized CA (certfificate authority) or we can create our own certificates using `javakey`. In our case we have generated our own certificate. A brief summary of the necessary procedures is as follows:

1. Create identity in the `javakey` database
2. Create signer in the `javakey` database
3. Generate public/private crpytographic keys for the signer
4. Create a certificate directive file
5. Generate a certificate associated with the signer
6. View the `javakey` database to ensure information is saved properly
7. Create an applet signing directive file
8. Create a jar file containing all applet resources
9. Sign applets using the applet signing directive file
10. Embed the signed applet into an HTML document using `OBJECT/EMBED` tags

The remainder of this document describes these steps in more detail. We should note that the 1.1.6 version of `javakey` appears to be broken. We have used the 1.1.5 version in our experiments.

1. First we create a trusted identity in the database:

   ```
   % javakey -c dcostello true
   ```

   This creates an identity named `dcostello` and sets "trusted"to true.

2. Next we inform the database that dcostello will be a trusted signer.

   ```
   % javakey -cs dcostello true
   ```

   This step will automatically create an `identity.obj` file in the home directory of the person executing this command. This seems like a bad idea since it means you would need multiple accounts to create different signers but this is the 1.1 model. We are exploring the new 1.2 model which solves some of these issues more intelligently.

   In any event, this will allow `dcostello` to sign applets and make them trusted to those who have the `identity.obj` file he provides them.

3. Now we can generate public/private key pairs for the new identity:

```
% javakey -gk dcostello DSA 512

Generated DSA keys (strength: 512).
```

By default `javakey` uses the DSA (Digital Signature Algorithm). Supposedly `javakey` can be instructed to use RSA if you have the licensing but we haven't tested this. We have found the DSA method with 512 bit keys, sufficient. Our goal is not the highest level of security but rather a reasonable way in which to get an applet permission to perform tasks that lie outside of the browser "sandbox security" model for Netscape and Internet Explorer.

`javakey` provides a way to save both your public and private keys to a file as well as other functionality. I will not discuss those things here but for more information see `http://java.sun.com/security/usingJavakey.html`.

4. Before applets can be signed we must have a certificate associated with the signer. An example certificate directive file is given in Figure C1. It appears there may be a Y2K problem with expiration dates (`end.date`).We haven't fully tested this but early tests seem to indicate trouble. Again the new signing tool should fix this (we can hope).
5. Now we can generate the certificate using the certificate directive file:

   ```
   % javakey -gc certdirective.txt
   ```
6. Next view the database to verify its contents.

   ```
   % javakey -ld
   ```

   The results should look something like the following:

   ```
   Scope: sun.security.IdentityDatabase, source file: /u/costello/id
   entitydb.obj

   [Signer]dcostello[identitydb.obj][trusted]

   public and private keys initialized

   certificates:

   certificate 1   for  : CN=dave costello, OU=cs department, O=Univ
   ersity of Rochester, C=United States

   from : CN=dave costello, OU=cs department, O=University of Roches
   ter, C=United States

   No further information available.
   ```
7. Now create the applet signing directive file. An example directive file is given in Figure C2.
8. Next create a jar file for the applet and it's resources:

   ```
   % jar cvf someapplet.jar somefile.class
   ```

   See the `jar` tool documentation for more info on using `jar`.
9. Then sign the applet using the applet directive file (see #7 above):

   ```
   % javakey -gs appletdirectivefile someapplet.jar
   ```

This step creates the new jar file containing the digital signature of the signer.

10. Embed the signed applet into an HTML document using `OBJECT/EMBED` tags. These tags are used to force the browser (specifically Netscape or Internet Explorer) to invoke the java-plugin. The plug-in is needed to allow the browser to recognize the applet as signed and trusted. Without the java-plugin the browser will not recognize a signed applet and therefore will not give it full permissions. An Example HTML document is shown in Figure C3.

    Notice that the `codebase` attribute, and the `pluginspage` attribute, are used to inform the browser that the plugin is needed. If the browser viewing the HTML document doesn't have the plug-in the user will be prompted to download and install the plug-in. The user is then taken directly to the plug-in download page. After installing the plug-in the user resumes the loading of the applet by clicking on a box displayed in the browser.

```
#
# Certificate Directive for javakey
# d costello Time-stamp: <98/10/27 16:09:41 costello>
# used for creating/issuing cryptographic certificates
#issuer
issuer.name=dcostello

#certifcate to use for signing (required if not self signed)
#issuer.cert=1

#required info
subject.name=dcostello
subject.real.name=dave costello
subject.org.unit=cs department
subject.org=University of Rochester
subject.country=United States

#cert info required
start.date=1 Oct 1998
end.date=30 Nov 1999
serial.number=1001

#signature algorithm to be used if not DSA
#signature.algorithm=MD5/RSA

#certificate file name
out.file=davecert.cer
```

**Figure C1: Example Certificate Directive File**

```
#
# JAR signing directive. This is the directive file used by javakey to
# sign a JAR file.
# dcostello Time-stamp: <98/10/27 17:07:01 costello>
#
# use:javakey -gs directivefile jarfile

# Which signer to use. This signer must be in the database.
     signer=dcostello

# Certificate number to use for this signer. This determines which
# certificate will be included in the PKCS#7 block. This is mandatory
# and is 1-based. Its value should be the number that javakey
# previously assigned to the signer's certificate when it generated it
# (or imported it). You can see which numbers javakey assigns
# to certificates by viewing the output of the
# -ld or -li javakey option.
        cert=1

# Certificate chain depth of a chain of certificates to include. This is
# currently not supported.
        chain=0

# The name to give to the generated signature file and associated signa-
ture
# block. This must be 8 characters or less.
# The generated signature file and associated signature block will have
# this name, with the .SF and .DSA extensions, respectively.
# In this example, the files will be DUKESIGN.SF and DUKESIGN.DSA.
        signature.file=DAVESIGN

# (Optional) The name to give to the signed JAR file.
        out.file=signedJar.jar
```

**Figure C2: Example Applet Signing Directive File**

```
<!-- HTML document containing an embedded java applet                    -->
<!--    The APPLET tags have been converted to OBJECT/EMBED tags          -->
<!--    to invoke the java plug-in inside the browser.                    -->
<html>
<body>
<!--"CONVERTED_APPLET"-->
<!-- CONVERTER VERSION 1.0 -->
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        WIDTH = 200 HEIGHT = 200
        codebase="http://java.sun.com/products/plugin/1.1.1/jinstall-111-
win32.cab#Version=1,1,1,0">
<PARAM NAME=CODE VALUE="KeybMgrApplet.class" >
<PARAM NAME=ARCHIVE VALUE="signedJar.jar" >
<PARAM NAME="type" VALUE="application/x-java-applet;version=1.1">
<PARAM NAME=geometry VALUE="80x4+0-0">
<PARAM NAME=serverhost VALUE ="mega.cs.rochester.edu">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.1"
       java_CODE="KeybMgrApplet.class"
       java_ARCHIVE="signedJar.jar"
       WIDTH=200 HEIGHT=200
       geometry="80x4+0-0"
       serverhost="mega.cs.rochester.edu"
       pluginspage="http://java.sun.com/products/plugin/1.1.1/plugin-
install.html">
<NOEMBED></COMMENT>
</NOEMBED></EMBED>
</OBJECT>
</body>
</html>
```

**Figure C3: Example HTML Document Using Signed Applet**

©U.S. GOVERNMENT PRINTING OFFICE: 2000-510-079-81275